

Za radoznale: Stanja igre i završni projekat

Ovo nije obavezno i neće biti na ispitu. Za one koji žele da vide kako se ista stvar radi naprednije. Svi ispisi su dobijeni pokretanjem u pygame-ce.

1. Stek scena: pauza „iznad“ igre

Na času je Pauza pamtila igru u svom atributu, da bi se vratila u nju. Veće igre često drže sve otvorene scene u listi koja se zove stek: nova scena se stavlja na vrh, a kad se zatvori, skida se i ispod nje je opet stara. Tako se pauza, pa pomoć iz pauze, pa nazad vraćaju bez ijednog posebnog atributa.

```
stek = ["meni"]

def otvori(scena):
    stek.append(scena)          # nova scena na vrh
    print("otvori", scena, "->", stek)

def zatvori():
    stara = stek.pop()         # skidamo scenu sa vrha
    print("zatvori", stara, "->", stek)

otvori("igra")
otvori("pauza")
otvori("pomoć")
zatvori()
zatvori()
print("sada je aktivna:", stek[-1])
```

Ispis:

```
otvori igra -> ['meni', 'igra']
otvori pauza -> ['meni', 'igra', 'pauza']
otvori pomoć -> ['meni', 'igra', 'pauza', 'pomoć']
zatvori pomoć -> ['meni', 'igra', 'pauza']
zatvori pauza -> ['meni', 'igra']
sada je aktivna: igra
```

Aktivna je uvek poslednja scena u listi, stek[-1]. Glavna petlja bi pozivala obradi i azuriraj samo za nju, a crtala bi sve scene redom, odozdo nagore, pa bi se zaleđena igra videla ispod pauze. Ovaj obrazac se u knjizi Game Programming Patterns zove „pushdown automata“ (poglavlje State).

Zašto ovo nismo radili na času: za četiri scene dovoljan je atribut sledeca; stek se isplati tek kad se scene otvaraju jedna iz druge u više nivoa.

Fajl: `primeri/za_radoznale/12_r1_stek_scena.py`

2. Najbolji rezultat koji se pamti i posle zatvaranja igre

Atribut klase Kraj.najbolji živi samo dok program radi. Da bi se rekord sačuvao i za sledeći put, upisujemo ga u tekstualni fajl. Funkcija open otvara fajl za čitanje ("r") ili za upis ("w"); blok with sam zatvara fajl.

```

import os

FAJL = "rekord.txt"

def procitaj_rekord():
    if not os.path.exists(FAJL):          # prvo pokretanje: fajla još nema
        return 0
    with open(FAJL) as f:
        return int(f.read())

def upisi_rekord(poeni):
    with open(FAJL, "w") as f:
        f.write(str(poeni))

for poeni in [7, 12, 9]:                  # tri partije
    if poeni > procitaj_rekord():
        upisi_rekord(poeni)
    print("partija:", poeni, "rekord:", procitaj_rekord())

```

Ispis:

```

partija: 7 rekord: 7
partija: 12 rekord: 12
partija: 9 rekord: 12

```

Fajl sadrži samo jedan broj kao tekst, zato pri čitanju koristimo int, a pri upisu str. Kad se program pokrene sutra, procitaj_rekord vraća 12, a ne 0.

Zašto ovo nismo radili na času: rad sa fajlovima nije deo ovog predmeta; za seminarski rad je ovo lep dodatak, ali nije obavezan.

Fajl: `primeri/za_radoznale/12_r2_rekord_u_fajlu.py`

3. Prelaz sa zatamnjenjem

Na času se scena menja u jednom frejmu. Lepši prelaz je kratko zatamnjenje: preko stare scene crtamo crnu površinu čija se neprovidnost (alfa, od 0 do 255) za pola sekunde poveća od 0 do 255, pa se tek onda pređe u novu scenu. Ovde samo računamo alfu za nekoliko trenutaka, bez prozora.

```

TRAJANJE = 0.5                                # sekundi
vreme = 0
dt = 0.1
while vreme < TRAJANJE:
    vreme += dt
    alfa = int(255 * min(1, vreme / TRAJANJE))
    print(round(vreme, 1), alfa)
print("sada prelazimo u novu scenu")

```

Ispis:

```

0.1 51
0.2 102
0.3 153
0.4 204
0.5 255
sada prelazimo u novu scenu

```

U igri bi ovo bila posebna scena Prelaz koja pamti staru i novu scenu: u azuriraj povećava vreme, u crtaj crta staru scenu i preko nje crnu površinu sa `set_alpha(alfa)`, a kad je alfa 255, postavlja sledeca na novu scenu. `min(1, ...)` ne dozvoljava alfu veću od 255.

Zašto ovo nismo radili na času: prelaz je ukras; dodaje još jednu scenu, a ne uvodi nov pojam.

Fajl: `primeri/za_radoznale/12_r3_prelaz_zatamnjenje.py`